1) Arbitrary Time Bucketing

Standard

```sql
SELECT
    to_timestamp(floor(open_time / 1000 / 900) * 900) AS bucket, -- 900s = 15m
    sum(volume) AS total_volume,
    max(high) - min(low) AS volatility
FROM data_bitcoin_control
GROUP BY bucket
ORDER BY bucket DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page No: 1

	bucket timestamp with time zone	total_volume double precision	volatility double precision
1	2024-12-31 23:45:00+00	61.03288	217.84999999999127
2	2024-12-31 23:30:00+00	99.67114000000001	278.9100000000035
3	2024-12-31 23:15:00+00	58.09506000000002	150.63999999999942
4	2024-12-31 23:00:00+00	117.78087	380.61999999999534

Total rows: 35136 | Query complete 00:00:14.196

Timescale

```sql
SELECT
    time_bucket('15 minutes', to_timestamp(open_time / 1000)) AS bucket,
    sum(volume) AS total_volume,
    max(high) - min(low) AS volatility
FROM data_bitcoin
GROUP BY bucket
ORDER BY bucket DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page No: 1

	bucket timestamp with time zone	total_volume double precision	volatility double precision
1	2024-12-31 23:45:00+00	61.032880000000006	217.84999999999127
2	2024-12-31 23:30:00+00	99.67114	278.9100000000035
3	2024-12-31 23:15:00+00	58.09505999999999	150.63999999999942
4	2024-12-31 23:00:00+00	117.78086999999998	380.61999999999534

Total rows: 35136 | Query complete 00:00:06.846

2) The "First & Last" Problem (Weekly OHLC)

Standard

```sql
SELECT
    date_trunc('week', to_timestamp(open_time / 1000)) AS bucket,
    (array_agg(open ORDER BY open_time ASC))[1] AS weekly_open,
    max(high) AS weekly_high,
    min(low) AS weekly_low,
    (array_agg(close ORDER BY open_time DESC))[1] AS weekly_close
FROM data_bitcoin_control
GROUP BY bucket
ORDER BY bucket DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 53 | Page No: 1

	bucket timestamp with time zone	weekly_open double precision	weekly_high double precision	weekly_low double precision	weekly_close double precision
1	2024-12-30 00:00:00+00	93738.19	96250	91530.45	93576
2	2024-12-23 00:00:00+00	95186.28	99963.7	92520	93738.2
3	2024-12-16 00:00:00+00	104463.99	108353	92232.54	95186.27
4	2024-12-09 00:00:00+00	101109.6	105250	94150.05	104463.99

Total rows: 53 | Query complete 00:00:01.831

Timescale

```sql
SELECT
    time_bucket('1 week', to_timestamp(open_time / 1000)) AS bucket,
    first(open, open_time) AS weekly_open, -- Native function
    max(high) AS weekly_high,
    min(low) AS weekly_low,
    last(close, open_time) AS weekly_close -- Native function
FROM data_bitcoin
GROUP BY bucket
ORDER BY bucket DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 53 | Page No: 1 of 1

	bucket timestamp with time zone	weekly_open double precision	weekly_high double precision	weekly_low double precision	weekly_close double precision
1	2024-12-30 00:00:00+00	93738.19	96250	91530.45	93576
2	2024-12-23 00:00:00+00	95186.28	99963.7	92520	93738.2
3	2024-12-16 00:00:00+00	104463.99	108353	92232.54	95186.27
4	2024-12-09 00:00:00+00	101109.6	105250	94150.05	104463.99

Successfully run. Total query runtime: 317 msec. 53 rows af

Total rows: 53 | Query complete 00:00:00.317 | CRLF | Ln 101, Col 1

3) Daily "Net Change"

Standard

```sql
SELECT
    date_trunc('day', to_timestamp(open_time / 1000)) AS day,
    (array_agg(close ORDER BY open_time DESC))[1] -
    (array_agg(open ORDER BY open_time ASC))[1] AS daily_change
FROM data_bitcoin_control
GROUP BY day
ORDER BY day DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 366 | Page No: 1

	day timestamp with time zone	daily_change double precision
1	2024-12-31 00:00:00+00	783.9499999999971
2	2024-12-30 00:00:00+00	-946.1399999999994
3	2024-12-29 00:00:00+00	-1561.800000000003
4	2024-12-28 00:00:00+00	1000.9700000000012

Total rows: 366 | Query complete 00:00:00.903

Timescale

```sql
SELECT
    time_bucket('1 day', to_timestamp(open_time / 1000)) AS day,
    last(close, open_time) - first(open, open_time) AS daily_change
FROM data_bitcoin
GROUP BY day
ORDER BY day DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 366 | Page No: 1

	day timestamp with time zone	daily_change double precision
1	2024-12-31 00:00:00+00	783.9499999999971
2	2024-12-30 00:00:00+00	-946.1399999999994
3	2024-12-29 00:00:00+00	-1561.800000000003
4	2024-12-28 00:00:00+00	1000.9700000000012

Total rows: 366 | Query complete 00:00:00.368

The "Gap Fill" Simulation (Interpolation)

Standard

```sql
SELECT
    series.bucket,
    (SELECT close FROM data_bitcoin_control
     WHERE open_time <= extract(epoch from series.bucket) * 1000
     ORDER BY open_time DESC LIMIT 1) as clean_close
FROM generate_series(
    '2024-06-01 00:00:00'::timestamp,
    '2024-06-02 00:00:00'::timestamp,
    '5 minutes'::interval
) AS series(bucket);
```

Data Output | Messages | Notifications

	bucket timestamp without time zone	clean_close double precision
1	2024-06-01 00:00:00	67573.41
2	2024-06-01 00:05:00	67640.69
3	2024-06-01 00:10:00	67620.77
4	2024-06-01 00:15:00	67553.41

Total rows: 289 | Query complete 00:00:55.800

Timescale

```sql
SELECT
    time_bucket_gapfill(
        '5 minutes',
        to_timestamp(open_time / 1000),
        start => to_timestamp(1717200000000 / 1000), -- Explicit Start (June 1)
        finish => to_timestamp(1717286400000 / 1000)  -- Explicit End (June 2)
    ) AS bucket,
    locf(last(close, open_time)) AS clean_close
FROM data_bitcoin
WHERE open_time BETWEEN 1717200000000 AND 1717286400000
GROUP BY bucket
ORDER BY bucket;
```

Data Output | Messages | Notifications

Showing rows: 1 to 289 | Page No: 1

	bucket timestamp with time zone	clean_close double precision
1	2024-06-01 00:00:00+00	67647.88
2	2024-06-01 00:05:00+00	67614.21
3	2024-06-01 00:10:00+00	67586.61
4	2024-06-01 00:15:00+00	67529.45

Total rows: 289 | Query complete 00:00:00.167